Or Group Design Project Report

Jacob Breiholz Emilio Esteban Gabriel Ritter

jsb4ns@virginia.edu eae4ct@virginia.edu gar5be@virginia.edu

ECE 3663 – Spring 2014  
University of Virginia

# INTRODUCTION

This paper details the design of 16 bit embedded digital signal processor for PICo using FreePDK 45nm technology. The ALU supports the seven required operations including NOP, ADD, SUB, SHIFT, AND, OR, and PASS. Additionally a multiplier function has been added.  Besides proving the functionality of the processor, this paper will provide metrics and explain design decisions made to improve them--primarily reducing delay, sizing, and power consumption.

# DESIGN DESCRIPTION

We implemented our design using following the format for a basic ALU.  The A and B inputs entered registers which proceeded to the ALU functions.  Afterwards, a mux controlled by the select lines determined which function to pass through to the final register.  A mux was additionally needed for the carry out which corresponded to the select lines for ADD or SUB.  We designed this, for the most part, using bitslicing techniques.  This involved designing everything for one bit and then stacking them to make it functional for 16 bits.  A few additional ANDs, ORs, and NOTs were necessary to do this.

The control points were chosen arbitrarily. If we were given information such as the probability that each operation would be called or the probability a that certain functions would be called after a specific function then the order could make a significant difference because of static power and capacitor charging and discharging upon transitions.

**Table 1. Operation Control Points**

|  |  |
| --- | --- |
| **Function** | **Control (“S2,S1,S0”)** |
| **ADD** | “000” |
| **SUB** | “001” |
| **SHIFT** | “010” |
| **AND** | “011” |
| **OR** | “100” |
| **PASS** | “101” |
| **Multiply** | “110” |
| **NOP** | “111” |

## Specific Design Decisions

The arbitrary function chosen was 8 bit by 8 bit multiplication because we felt it would add substantial functionality to the processor.

NOP was done by routing the output back into the corresponding NOP input of the ALU mux.

PASS utilizes transmission gates rather than pass gates because although this results in a greater size we believed it to be negligible and thought that the advantages of the gate pulling the voltages all the way up and down was more desirable.

## 3. INNOVATION

Many innovations were implemented to reduce delay, sizing, and power consumption based off of the metric equation. Our primary focus was designing the processor by making intelligent decisions for the implemented topologies.

## 3.1 Adder and Subtractor

The one-bit adder was a block which was used in multiple portions of the project, including the 16 bit adder, 16 bit subtractor, and multiplier blocks. It was therefore critical that an efficient design be chosen for this recurring device. We implemented a mirror adder topology to try to reduce delay as simply and effectively as possible. This adder is much more efficient at propagating carries than a normal adder and with very little increase in size.  There are other more efficient adders in terms of delay, however they are also much larger in size and complexity.  Although size is not as important a factor as delay, we took the complexity into consideration based on the size of our team and the amount of time we were able to devote to this project.   The subtractor was implemented the same way as the adder but the B input was inverted and a carryout was added.  This converts B into a 2’s complement number which is like adding a negative number or in other words, subtraction.

## 3.2 Multiplier

A Wallace Tree multiplier was chosen to implement the arbitrary function. The multiplier designed used a total of 14 half adders and 39 full adders in the compression stages alone, with an additional as 9 full adders and 2 half adders for the fast adder on the end. This total of 64 adders is slightly larger than the number that would be necessary in an equivalent array multiplier. However, the speed gained from using a Wallace Tree multiplier will help the decidedly more complicated block perform at the same clock speeds as the simpler operations on the ALU. For simplicity’s sake, no half adder block was designed. The number of half adders used was deemed small enough that there would be little benefit from designing another logic block when the same function could be accomplished by tying an input of a regular adder to zero. Furthermore, in some cases the half adder is expected to receive a carry bit. Using the original mirror adder block as discussed above allowed for greater flexibility when optimizing of the path.

## 3.3 Shifter

The shifter is comprised almost entirely of single bit 2:1 muxes.  It has a total of 48 of these muxes laid out in three columns of 16--one for each bit. The first column was designed to shift one bit, the second column two bits, and the third column could have been designed to shift either two or four bits to satisfy the requirements. The requirements were to be able to shift 1, 2, 3, or 4 bits.  The first three are taken care of by combinations of the first two columns and shifting four bits could be accomplished by either using the third column to shift four bits or by using it to shift two bits or one bit and shifting in combination with another column.  We chose four bits because that would allow greater variety in possible shifts if a greater number of shifts was ever desired.  Furthermore in the shifter design, since there were three columns and two select inputs, some minor combinational logic had to be implemented to apply signals to the columns. Because of the sheer number of muxes, they were designed using transmission gate logic to reduce the size and power consumption at a minor cost to delay in this case.

## 3.4 AND and OR

As a whole in order to reduce power and delay we made an effort to reduce glitching in our multi-bit building block components (AND and OR).  This was done by designing them such at all intermediate nets reached the final stage of the component at the same time.   This was important since these basic components were using over and over throughout the design.

# 3.5 Sizing

Our final step to optimize delay was the use logical effort to resize the components.  For this the clock buffer size needed to be assigned arbitrarily and we started off with a value of 20.  After sizing based off of that, we found delay to be limited by the clock’s rising and falling edges.  Therefore the clock’s buffer size was messed around with until we found that at higher buffer sizes the delay achieved by resizing of components was detrimental to the metric since the size value exceeded the delay squared value.  This lead us to leave the default component sizes and make the clock buffer large, since a higher clock buffer would mean faster transitions and less delay at no cost to the metric.  The clock buffer size was finalized at a size of 1000x the size of the characteristic inverter for each of the two inverters of the buffer.

# RESULTS

The following detail the metrics of the designed processor.

## Area

The area was found by finding the size of each component compared to the characteristic inverter.  For the entire processor, the size is 3730 times the size of the characteristic inverter.  To compute the area, the size was then multiplied by the width of the characteristic inverter which had pmos and nmos widths of 90 nm.  The final result of the area is 6.714\*10-4 m.  The area of the multiplier is 2725 characteristic inverters or 4.905\*10-4 m.

## Delay

The delay was found by creating the worst case scenario for the adder, which when tested in earlier stages of the alu design was shown to have the worst delay out of all the components.  Then, simulation was done for this case while gradually reducing the clock cycle to the point where there was no longer functionality.  This gave us our worst case delay as well as maximum clock frequency.  After these were obtained the other operations were tested at the same frequency to confirm they worked as well.  The delay is 625 ps and therefore the clock frequency is 1.6 GHz.

**Table 2. Worst Case Delays of Processor and Components**

|  |  |
| --- | --- |
| **Component** | **Delay (ps)** |
| **ADD** | 522 |
| **SUB** | 495 |
| **AND** | 14.4 |
| **OR** | 14.6 |
| **PASS** | 15.1 |
| **SHIFT** | 354 |
| **Multiplier** | 297.9 |
| **Processor** | 625 |

## Energy

Average energy was calculated as instructed in the requirements. Displayed is the power instead of energy since it is more applicable to real situations.

**Table 3. Average Power Use of Processor and Components**

|  |  |
| --- | --- |
| **Component** | **Power (W)** |
| **ADD** | 4.904\*10-5 |
| **SUB** | 4.691\*10-5 |
| **AND** | 6.685\*10-6 |
| **OR** | 1.665\*10-5 |
| **PASS** | 2.299\*10-11 |
| **SHIFT** | 3.852\*10-6 |
| **Mux** | 1.572\*10-5 |
| **Register** | 1.115\*10-4 |
| **Multiplier** | 2.450\*10-5 |
| **Processor** | 2.503\*10-4 |

## Metric

The metric of our design is area\*delay^2\*power.  This comes out to be 6.56\*10^-26 m\*s^2\*W

# Conclusion

Our design offers the full functionality desired while optimizing the metric for size, delay, and power consumption.  Additionally, the multiplier feature has been added. First and foremost we made sure of the functionality of the ALU.  Through rigorous testing, simulation, and redesigning we arrived at a completely functional and stable result which can be seen in our supplemental documents.  Besides this, we implemented many innovations to our design within reason of the limited time and team we had available to produce the best metric possible.  Having discussed these in depth in this report and having presented the metrics requested, we hope PICo sees the merits and superiority of our product.